CS M151B Homework 7
Baolinh Nguyen
UID: 104732121

5.3 For a direct mapped cache design with a 32-bit address, the following bits of the address are used to access the cache.

Tag	Index	Offset
31-10	9-5	4-0

5.3.1 [5] <§5.3> What is the cache block size (in words)?

5.3.2 [5] <§5.3> How many entries does the cache have?

Starting from power on, the following byte-addressed cache references are recorded.

0	4	16	132	232	160	1024	30	140	3100	180	2180

5.3.4 [10] <§5.3> How many blocks are replaced?

5.3.5 [10] <§5.3> What is the hit ratio?

5.3.1. $5 = \log_2(\text{block size})$

block size $= 2^5 = 32$ bytes $\Rightarrow \boxed{8 \text{ words per block}}$

32 bits/word $\Rightarrow$ 4 bytes/word

5.3.2. $2^5 = \boxed{32 \text{ entries}}$

5.3.4.

	Tag [31-10]	Index [9-5]	Offset [4-0]	Result
0	0	00000	00000	Miss
4	0	00000	00100	Hit
16	0	00000	10000	Hit
132	0	000100	00100	Miss
232	0	00111	01000	Miss
160	0	00101	00000	Miss
1024	1	00000	00000	Miss - b/c of tag bit (0) replace
30	0	00000	11110	Miss - b/c of tag bit replace (1)
140	0	00100	01100	Hit
3100	11	00000	11100	Miss - b/c of tag bit (0) replace
180	0	00101	10100	Hit
2180	10	00100	00100	Miss - b/c of tag bit (0) replace

$\boxed{4 \text{ blocks replaced}}$

5.3.5 Hit ratio $= \frac{4 \text{ hits}}{12 \text{ instr}} = \boxed{33.3\%}$

CS M151B Homework 7
Baolinh Nguyen
UID: 104732121

5.5 Media applications that play audio or video files are part of a class of workloads called "streaming" workloads; i.e., they bring in large amounts of data but do not reuse much of it. Consider a video streaming workload that accesses a 512 KiB working set sequentially with the following address stream:

0, 2, 4, 6, 8, 10, 12, 14, 16, ...

5.5.1 [5] <§§5.4, 5.8> Assume a 64 KiB direct-mapped cache with a 32 byte block. What is the miss rate for the address stream above? How is this miss rate sensitive to the size of the cache or the working set? How would you categorize the misses this workload is experiencing, based on the 3C model?

5.5.2 [5] <§§5.1, 5.8> Re-compute the miss rate when the cache block size is 16 bytes, 64 bytes, and 128 bytes. What kind of locality is this workload exploiting?

5.5.1 capacity of cache $\frac{64 \times 1024 \text{ B}}{32 \text{ B / block}} = 2048$ blocks

Since there are 2048 blocks/slots, there will be $\log_2(2048) = 11$ index bits.
Since each block is 32 bytes, there will be $\log_2(32) = 5$ offset bits
A block can hold up to 32 of these addresses, and since we access every 2nd address, we can store up to $\frac{32}{2} = 16$ addresses in a single block. Thus, though the first access of a set of 16 addresses will be a miss, the remaining 15 will be hits. Thus, our miss rate is $\frac{1}{16} \cdot 100 = \boxed{6.25\%}$ The miss rate is sensitive to the size of the blocks in the cache and these are compulsory misses since they are misses that occur due to never having seen that address before.

5.5.2. In general, the miss rate $= \frac{\text{jump size}}{\text{block size}}$.

16 bytes $\Rightarrow \frac{2}{16} \times 100 = \frac{1}{8} \times 100 = \boxed{12.5\%}$

64 bytes $\Rightarrow \frac{2}{64} \times 100 = \frac{1}{32} \times 100 = \boxed{3.125\%}$

128 bytes $\Rightarrow \frac{2}{128} \times 100 = \frac{1}{64} \times 100 = \boxed{1.5625\%}$

Because we are accessing addresses that are close to each other, this is taking advantage of $\boxed{\text{spatial locality}}$

CS M151B Homework 7

Baolinh Nguyen

UID: 104732121

5.6 In this exercise, we will look at the different ways capacity affects overall performance. In general, cache access time is proportional to capacity. Assume that main memory accesses take 70 ns and that memory accesses are 36% of all instructions. The following table shows data for L1 caches attached to each of two processors, P1 and P2.

[illegible]	[illegible]	[illegible]	[illegible]
P1	2 KiB	8.0%	0.66 ns
P2	4 KiB	6.0%	0.90 ns

5.6.2 [5] <§5.4> What is the Average Memory Access Time for P1 and P2?

For the next three problems, we will consider the addition of an L2 cache to P1 to presumably make up for its limited L1 cache capacity. Use the L1 cache capacities and hit times from the previous table when solving these problems. The L2 miss rate indicated is its local miss rate

[illegible]	[illegible]	[illegible]
1 MiB	95%	5.62 ns

5.6.4 [10] <§5.4> What is the AMAT for P1 with the addition of an L2 cache? Is the AMAT better or worse with the L2 cache?

5.6.2.

$$AMAT_1 = \frac{1(0.66+0.08(70)) + 0.36(0.66+0.08(70))}{1.36} = \boxed{6.26ns}$$

$$AMAT_2 = \frac{1(0.90+0.06(70)) + 0.36(0.90+0.06(70))}{1.36} = \boxed{5.1ns}$$

5.6.4.

$$AMAT_1 = \frac{1(0.66+0.08(5.62+0.95(70))) + 0.36(0.66+0.08(5.62+0.95(70)))}{1.36}$$

$$= \boxed{6.4296\ ns}$$

$\boxed{\text{The AMAT is worse w/ the added L2 cache}}$

5. ***Why, oh why, must we do TCPI? (40 points)***: We are going to assess branch and cache performance on the pipelined datapath from class – we have full data forwarding. Our peak CPI is 1.0. Assume that 30% of instructions are branches, and that we have a single cycle branch hazard on this processor. Our branch predictor **always** guesses *not taken*. 50% of branches are not taken. Our processor has an instruction cache and data cache – both take a single cycle to access. The instruction cache miss rate is 10% and the data cache miss rate is 30%. The next level of the memory hierarchy is an L2 cache with a miss rate of 20% and an access time of 10 cycles, this is in addition to the L1 cache latency. Main memory has an access time of 80 cycles, this is in addition to the latency of the L1 and L2 caches. 20% of instructions are loads, and stores do not stall the processor on a cache miss. 3/5ths of loads have dependent instructions following them. Our target application executes 1,000,000 instructions. The processor clock runs at 2 GHz.

a. Calculate the average memory access time (AMAT) of the processor:

10% 30%

$$\begin{tikzcd} I\$\ 1 \arrow[dr, dash] & & D\$\ 1 \arrow[dl, dash] \\ & L2 \arrow[d, dash] & 20\%\ \ -10 \\ & MEM\ \ 80 & \end{tikzcd}$$

AMAT: 4.467 cycles

$$AMAT = \underbrace{1(1+0.1(10+0.2(80))) + 0.2(1+0.3(10+0.2(80)))}_{1.2}$$

$= 4.467$ cycles

b. Calculate TCPI for our target application on our processor.

TCPI: 5.43

$TCPI = BCPI + MCPI$

$BCPI = CPI_{peak} + CPI_{data\ hazard} + CPI_{control\ hazard}$

$= 1.0 + (0.2)(0.6)(1) + (0.3)(0.5)(1) = 1.27$

$MCPI = 0.1(10+0.2(80)) + 0.2(0.3(10+0.2(80)) = 4.16$

$TCPI = 1.27 + 4.16 = 5.43$

c. Suppose 1/6th of all branches are procedure calls. Each procedure call (i.e. a jal instruction) in our application also has a return (i.e. a jr instruction). These will all be mispredicted because we always guess not taken. One approach to reducing branch hazards in such a case is to *in-line* the procedure call. The compiler basically takes the instructions in the body of the procedure call and replaces **all** calls to that procedure with these instructions. This means that instead of the code:

```
        add $s0, $s0, $t1
        jal Target
        add $s0, $s0, $t2
        jal Target
        .....
        .....
        .....
        .....
Target: lw $t3, 0 ($s0)
        addi $t3, $t3, 200
        sw $t3, 0 ($s0)
        jr $ra
```

We would have the code:

```
add $s0, $s0, $t1
lw $t3, 0 ($s0)
addi $t3, $t3, 200
sw $t3, 0 ($s0)
add $s0, $s0, $t2
lw $t3, 0 ($s0)
addi $t3, $t3, 200
sw $t3, 0 ($s0)
.....
.....
.....
.....
```

The benefit in this simple example is that we avoid four branches (two jal's and two jr's), but the size of the instruction text segment in memory (i.e. the size of the actual program we are running) has increased. Now, instead of the lw, addi, and sw being in one place in the text segment, they are in two places. This can increase the miss rate of the instruction cache.

Suppose that we try in-lining on our processor. In order for performance to improve, the cost of increasing the instruction cache miss rate must not exceed the benefit of reducing branch hazards. Using TCPI as the CPI in the equation for Execution Time, provide an upper bound on the miss rate of the instruction cache to improve performance when using in-lining. Assume that the L2 cache's miss rate does not change.

The instruction cache miss rate must be <= 12%

We have 1000000 instructions. By inlining, we can reduce the number of branch instructions.

$IC_{branch} = (100,000)(0.30)\left(\frac{6-2}{6}\right) = 20,000$ instructions.

Previously, we had 50% of branches as predicted not taken. We've removed 10000 instructions (the jal/jr) that were predicted incorrectly so now, we have a miss rate of

$MR_{branch} = \frac{150000}{200000} - \frac{100000}{200000} = \frac{50000}{200000} = 0.25$

Thus

$ICPI_{new} = (1.0 \times MR_I(10 + 0.2(80))) + 0.2(0.3(10 + 0.2(80)))$

$= 26\ MR_I + 1.73$

$BCPI_{new} = 1.0 + (0.2)(0.6)(1) + \frac{200000}{1000000}(0.25)(1) = 1.18$

$TCPI_{new} = 1.18 + 26 MR_I + 1.73$

Looking at our instruction count,

$IC_{new} = 100000 - \frac{1}{3}(0.3)(100,000) = 900000.$

$ET_{old} = 5.43 \times 1000000 \times \frac{1}{2 \times 10^9} = 0.002715$

$ET_{new} = (1.18 + 26 MR_I + 1.73)(900000)\left(\frac{1}{2 \times 10^9}\right)$

$= 0.00131 + 0.0117 MR_I$

$ET_{old} \geq ET_{new}$

$0.002715 \geq 0.00131 + 0.0117 MR_I$

$MR_I \leq 0.1201 \Rightarrow 12\%$

$MR_I \leq 12\%$